ThomasCarstens / **SelfStudyIRM**

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

master

SelfStudyIRM / OS_and_Programming / **main.py** / Jump to

ThomasCarstens format

1 contributor

Raw | Blame

248 lines (199 sloc) 4.45 KB

Code navigation is available!

Navigate your code with ease. Click on function and method calls to jump to their definitions or references in the same repository. Learn more

```
import numpy as np
import math
import test
import matplotlib.pyplot as plt
import statistics#a importer

def openfile():
    fh = open("testFile.txt", 'rw')
    fh.write("dslfk")
    fh.close()
    #with open("test2File", 'rw') as fh:
    #    fh.write("ooo")
    a = 'hello \"world\"'
    print (a)

def helloworld():
    print ("Hello World")
#helloworld()


def listints():
    for i in range(101):
        print i

def userinput():
    a = input()
    print a

def userreps():
    reps = input()
    #print reps
    for i in range(reps):
        print ("HQ")
#userreps()

def logvalue():
    value = input()

    logvalue = math.log(value,10)
    print logvalue

#logvalue()

def regurgitate():
    store = []
    a = 0
    if not a:
        a = input()
        store.append(a)
        print(store)
    print(store) #???????
    for i in store.size():
        print(store.pop())
regurgitate()


def patt():
    store = {}
    a = 0
    for i in range(5):
        a[i] = input()
        #store.append(a)
        print(store)
    print(store) #???????
    for i in range(store.size()):
        print(store.pop())
#patt()
# if 1 in a: etc.

#def sort():
    #a.sort() #sort se fait "in-place"

#a=input.lower()
```

```
for char in a:
    if char in freq
        freq[char += 1
        print(char)
    else:


#Write a program that checks of two lists
#have at least 1 element in common

#LIST OF CHARACTERS TO STRING.
"".join(a)
"-".join(a)

#TUPLES
def f(*arg): # kargs
    print(arg)
# arguments sont tuples avant de rentrer dans
# la fonction.
f(1,2)

#PAR DEFAUT
def f(arg1, arg2=3):
    ...

#DICTIONNAIRE D'ARGUMENTS
#needs looking into
f(1, name=3, ok=True)
f(1, name=3)

#Write a function that takes an arbitrary amount of arguments and returns the sum of
#these arguments.
#sum
#moyenne
#stddev
def sum(*args):
    return sum(args)

def f(*args)
    return sum(args, statistics.mean(args, statistics.stdebv(args)))

import json
#from PYTHON DICT TO JSON
a = {
    'a': 42
}
s = json.dumps(a)
    disc_python=json.loads("""
    {
        a:1
    }
    """)


#TRY-EXCEPT BLOCK: TO MANAGE ERROS
try:
    a=1/0
except Exception as err:
    print("On a eu une erreure.")
    #optionel: print(err)
##########################################3
A VERIFIER: CLASSE
class Maison:
    def __init__(self, nb_pieces):
        self.nb_pieces = nb_pieces
        self.record = {
                        "name": 3,
                        "balance": 100,
                        "transaction":[100]
                        }

    def __getitem__(self, key):
        if key in self.record:
            return self.record[key];
            print (self.record[key])
        else:
            return "No such category."

    def __setitem__(self, key, value):

        if key in self.record:
            self.record[key]= value
        else:
            return "No such category."

    def afficher_nb_pieces(self):
        print(self.nb_pieces)

m1 = Maison(3)
#print(m1.nb_pieces)
#m1.afficher_nb_pieces()
m1[0]= "oui"
m1[2]

########################################


GENERATEUR: garder l'etat interne fonction
def f():
   i=0
   while True:
       yield i #conserve etat interne fonction
       i += 1
gen = f()
print(next(gen))
```

```
print(next(gen))


return a if var else b
dont l'exemple avec if .. in ..

NEW WAY OF DISPLAYING LOOPS
l = [i for i in range(10)]
[]<var> for <var> in <iter> (condition)]

#GREGOR
class NoneDico:
def __init__(self):
self.dico = {}

def __getitem__(self, key):
return self.dico[key] if key in self.dico else None

def __setitem__(self, key, value):
self.dico[key] = value

#GREGOR
class Readable():
def __init__(self, nb_pages, title, author="Unknown"):
self.nb_pages = nb_pages
self.title = title
self.author = author

def read(self):
print("Reading "+str(self.nb_pages)+" pages from "+self.author)

class Book(Readable):
def __init__(self, nb_pages, title, author):
super().__init__(nb_pages, title, author)

class Magazine(Readable):
def __init__(self, nb_pages, title):
super().__init__(nb_pages, title)

b = Book(42, "L'assomoir", "Zola")
m = Magazine(111, "Voici")
b.read()
m.read()

#GREGOR
import math
def f(i):
i = math.log(i) + math.sqrt(i)
i /= 24
i = i << 2
return i


l = [f(i) for i in range(10)]

#[<var> for <var> in <iter> (condition)]


####
def intervalles():
    #https://numpy.org/doc/stable/reference/generated/numpy.linspace.html
    evenly=np.linspace(0, 100)
    print(evenly)
#intervalles()

def graph():
    for i in range (100):
        x1 = i
        y1=2*x1
        plt.plot(x1, y1, 'bo')
        plt.show()
graph()
```